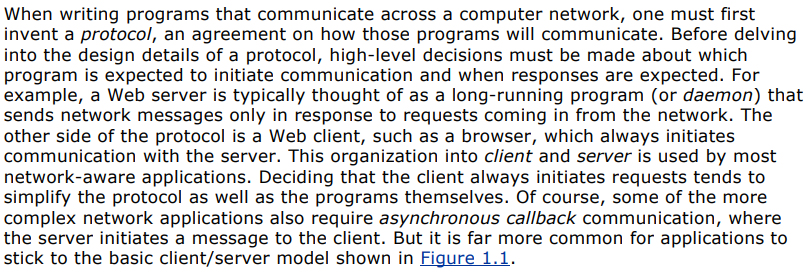
UNIT 1

1. What is network protocol? With a neat block diagram explain the network application for client and server.

A protocol is an established set of rules that determines how data is transmitted between different devices over the network.

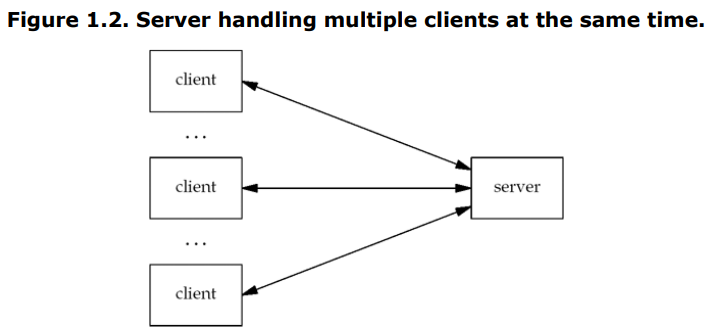


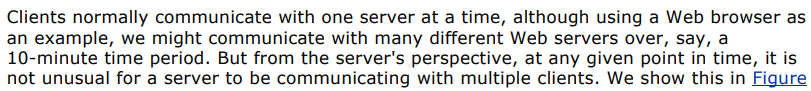


Client is a program (process) that sends requests to the server for some information. For ex: Browser

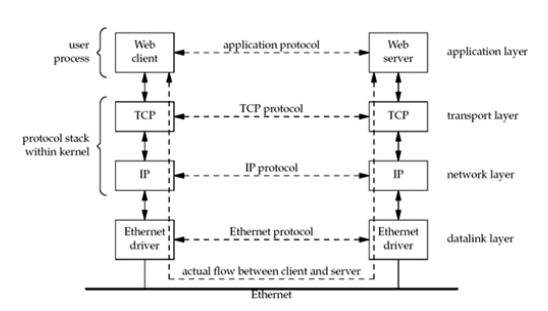
Server is a program (process) that sends information to client only in response to coming in requests from network. For ex: Web server

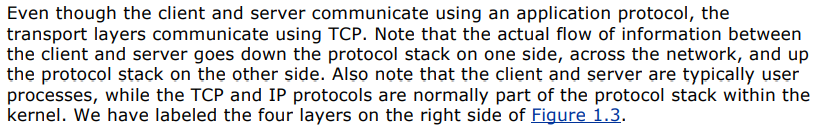
1. List out the various approaches used to handle multiple clients at the same time.



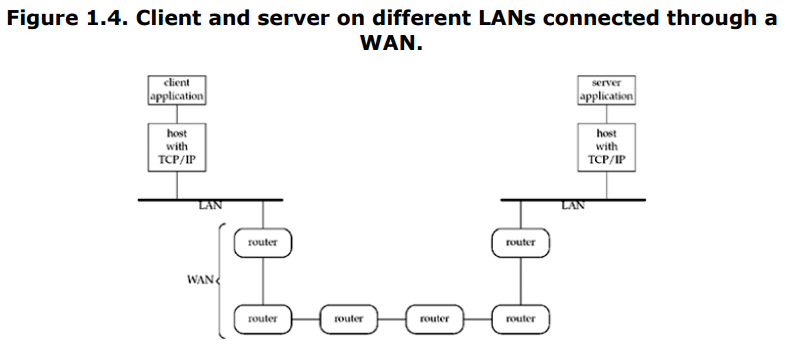


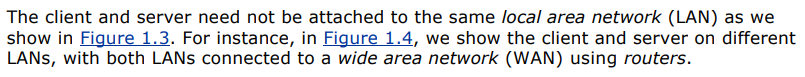
1. With a neat block diagram, explain the client and server communication on Local Area Network using TCP.

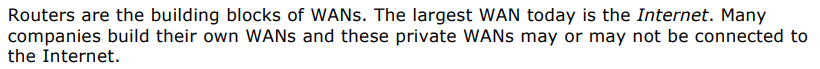




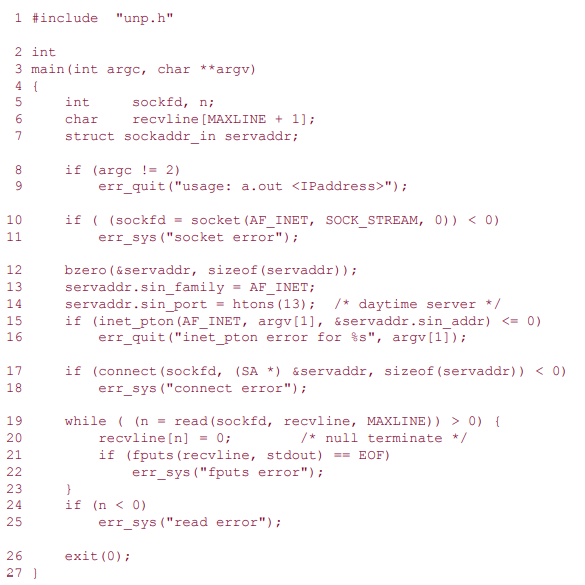
1. With a neat block diagram, explain the client and server communication over Wide Area Network using TCP.



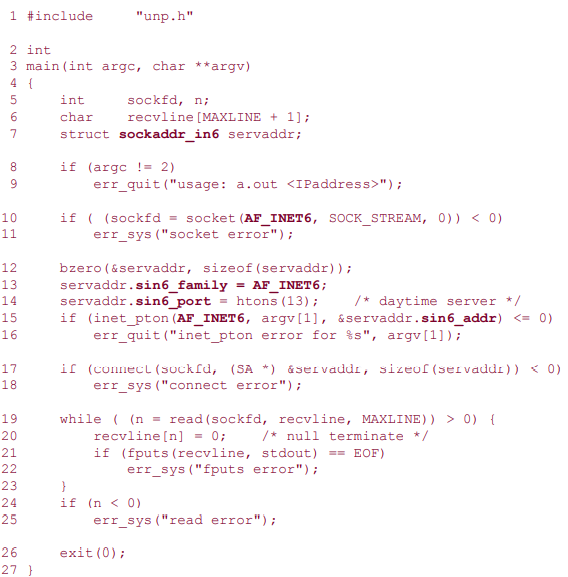




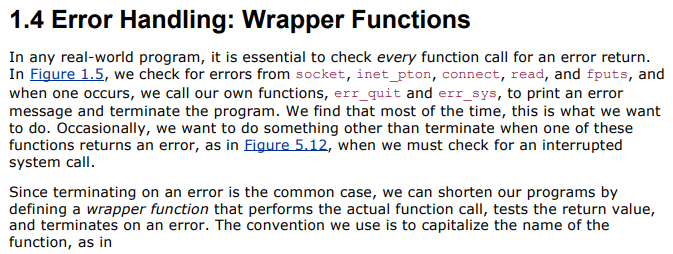
1. List and explain the steps involved in simple daytime client.
2. Include header
3. Command line arguments
4. Create Tcp socket
5. Specify servers IP address and Port
6. Establish a connection
7. Read and display servers reply
8. Terminate the program
9. Develop the ‘C’ program to implement simple daytime client.



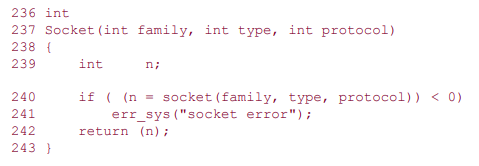
1. Comment on the Protocol Independence. Modify the day time client program for IPv6.



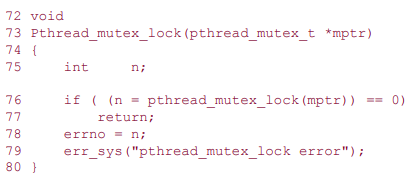
1. What are wrapper functions? Develop the wrapper function for the following:



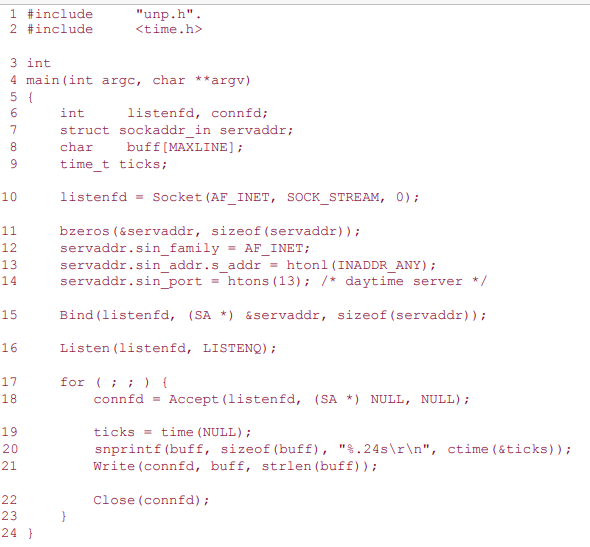
* 1. Socket function



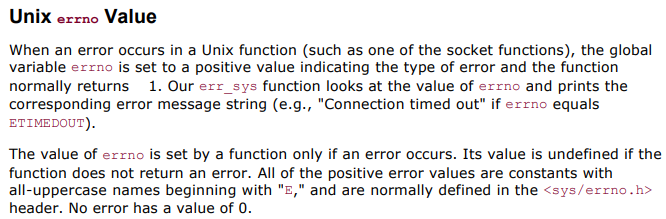
* 1. Pthread\_mutex\_lock



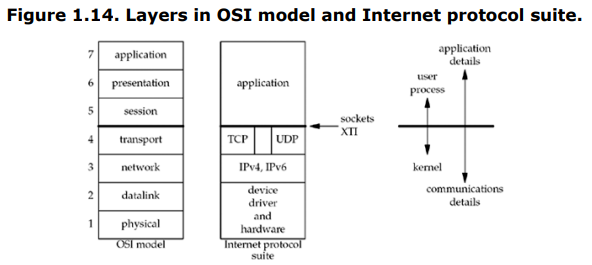
1. List and explain the steps involved in simple daytime server.
2. Create a TCP Socket
3. Bind the servers well Known port to socket
4. Convert socket to Listening socket
5. Accept clients connection and reply
6. Terminate program
7. Develop the ‘C’ program to implement simple daytime server.



1. Write a note on Unix errno value.

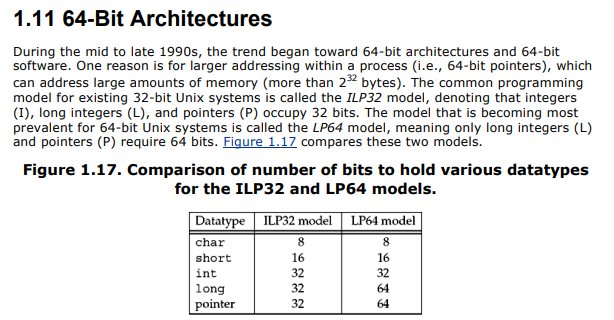


1. Explain with a neat block diagram the layers of OSI model and Internet protocol suite.

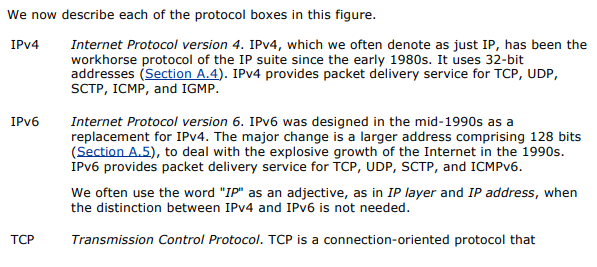


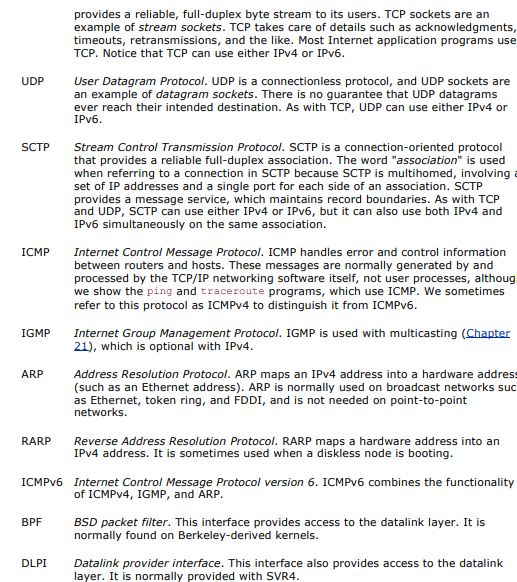
* We consider the bottom two layers of the OSI model as the device driver and networking hardware that are supplied with the system.
* The network layer is handled by the IPv4 and IPv6 protocols
* We show a gap between TCP and UDP in Figure 1.14 to indicate that it is possible for an application to bypass the transport layer and use IPv4 or IPv6 directly. This is called a raw socket,
* The upper three layers of the OSI model are combined into a single layer called the application. This is the Web client (browser), Telnet client, Web server, FTP server, or whatever application we are using. With the Internet protocols, there is rarely any distinction between the upper three layers of the OSI model.
* Why do sockets provide the interface from the upper three layers of the OSI model into the transport layer? There are two reasons for this design, which we note on the right side of Figure 1.14.
* First, the upper three layers handle all the details of the application (FTP, Telnet, or HTTP, for example) and know little about the communication details. The lower four layers know little about the application, but handle all the communication details: sending data, waiting for acknowledgments, sequencing data that arrives out of order, calculating and verifying checksums, and so on.
* The second reason is that the upper three layers often form what is called a user process while the lower four layers are normally provided as part of the operating system (OS) kernel. Unix provides this separation between the user process and the kernel, as do many other contemporary operating systems. Therefore, the interface between layers 4 and 5 is the natural place to build the API.

1. Write a note on 64- bit architectures.

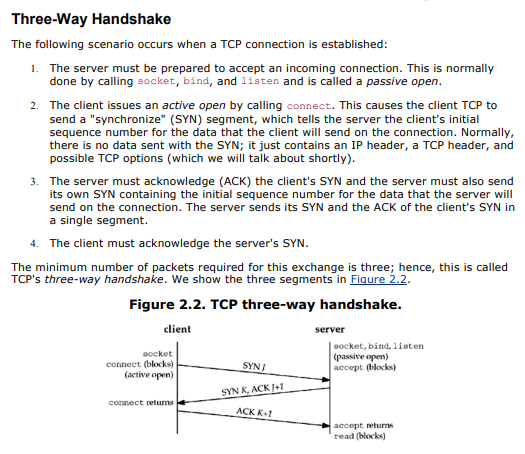


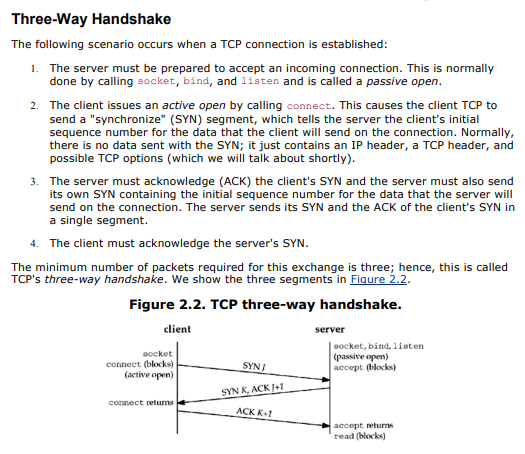
1. Explain the features of the following protocols:
   1. IPv4
   2. IPv6
   3. TCP
   4. UDP
   5. SCTP
   6. ICMP
   7. IGMP
   8. ARP
   9. RARP
   10. ICMPv6
   11. BPF
   12. DLPI

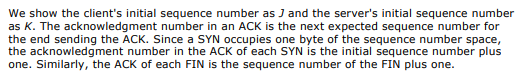




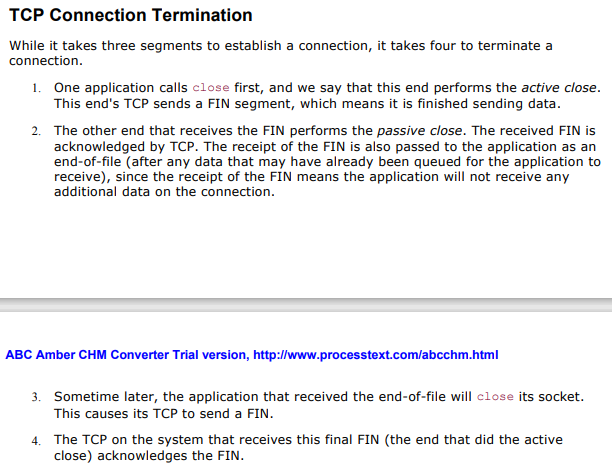
1. List and explain the features of UDP Protocol in detail.
2. List and explain the features of TCP Protocol in detail.
3. Explain with a neat diagrams the following:
   1. TCP connection establishment

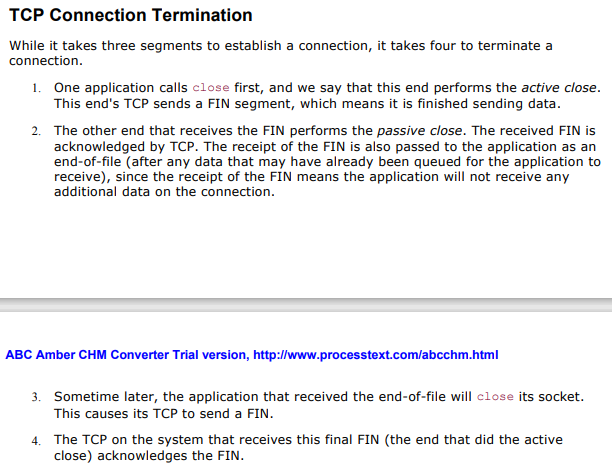


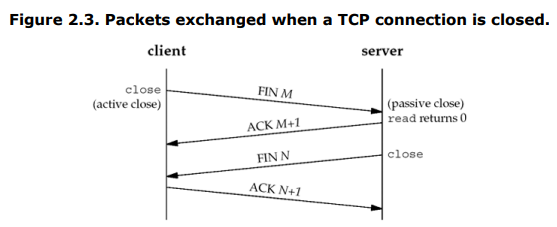




* 1. TCP data transfer
  2. TCP connection termination





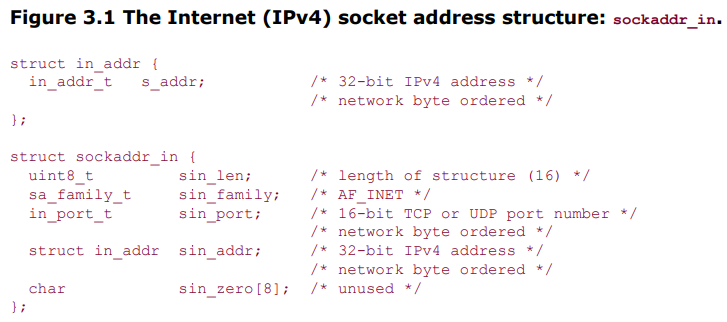


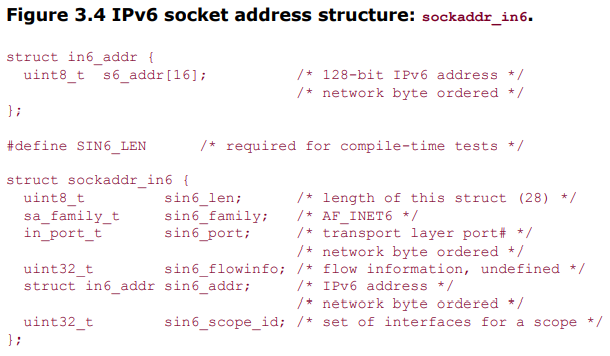
1. Explain the TCP State Transition diagram with a neat diagram.

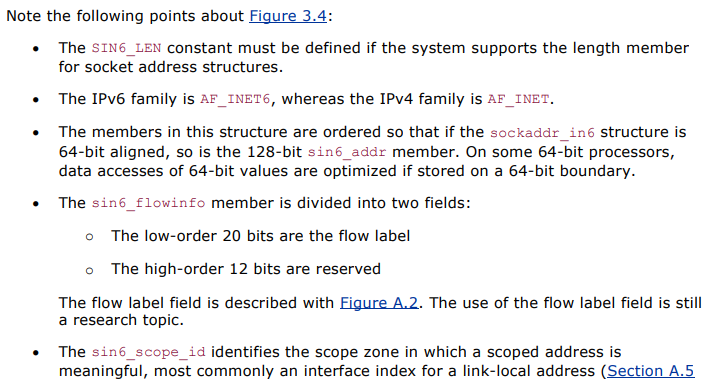
Unit 2: (Sockets Introduction)

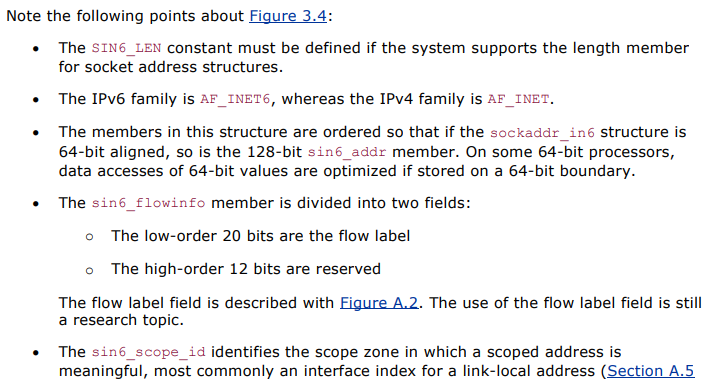
1. With a standard POSIX definition explain socket address protype for IPv4 *(sockaddr\_in)* and IPv6 *(sockaddr\_in6)*

POSIX (Portable Operating System Interface) is a set of standard operating system [interfaces](https://www.techtarget.com/whatis/definition/interface) based on the [Unix](https://www.techtarget.com/searchdatacenter/definition/Unix) operating system.

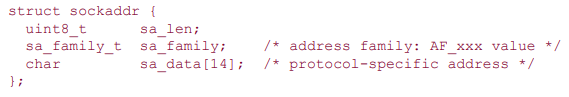




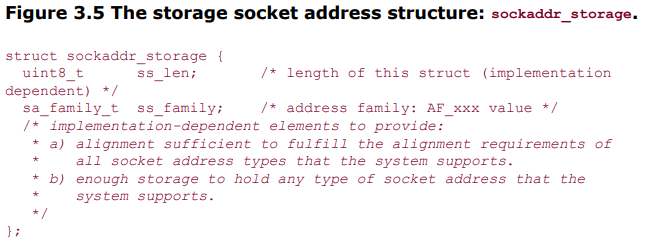




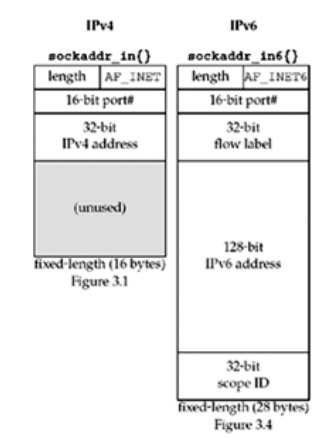
1. With a standard POSIX definition explain socket address structure protype: *sockaddr*



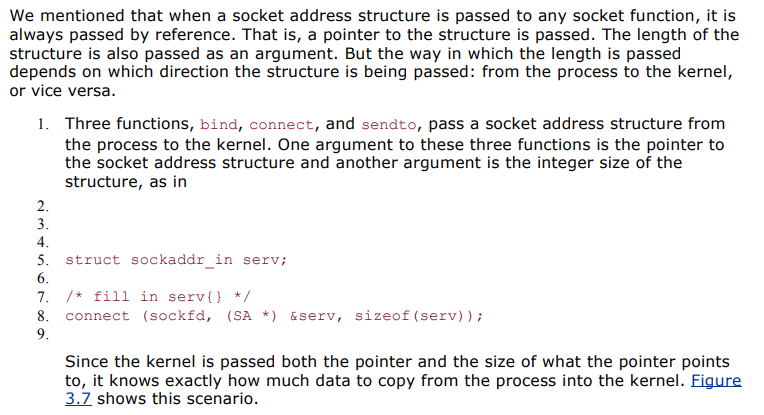
1. Show the prototype for storage socket address structure: *sockaddr\_storage*

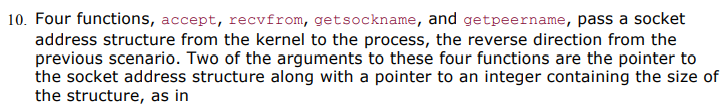


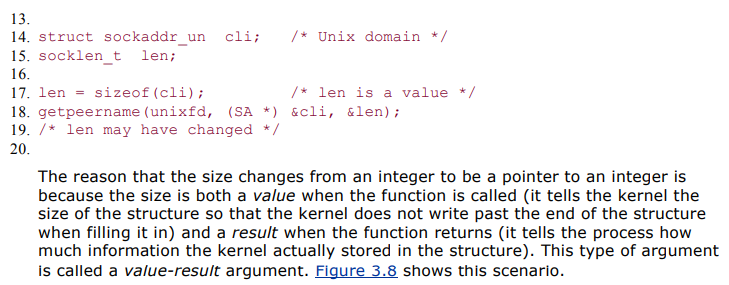
1. Compare the various socket address structures: *sockaddr\_in (), sockaddr\_in6.*

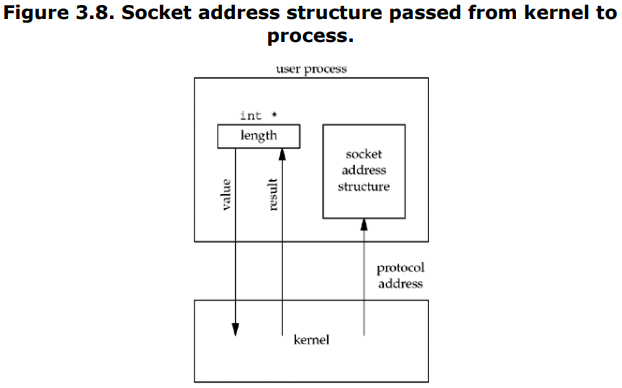


1. What are Value- Result Arguments? Explain the scenario with a neat block diagram.

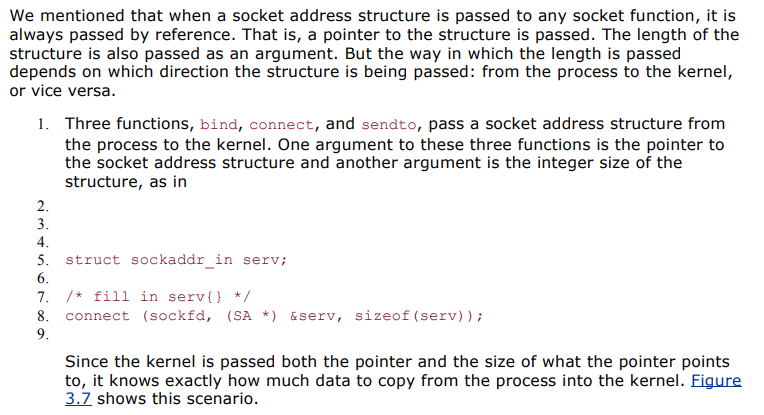


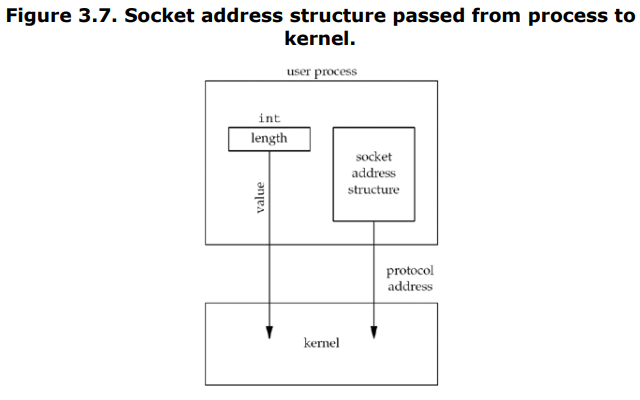




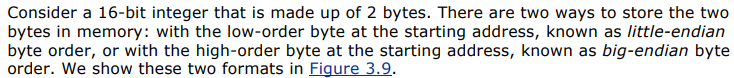


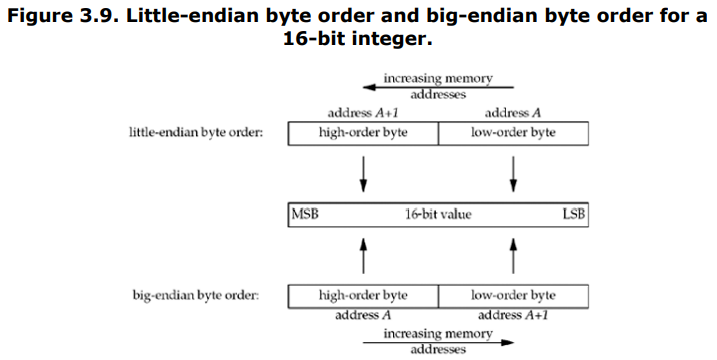
1. Explain the functions which passes socket address structure from the process to the kernel with a neat block diagram.

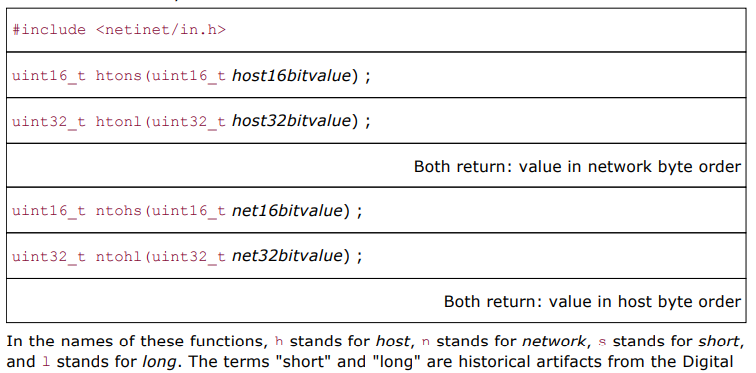




1. Explain with a neat diagram the various byte ordering functions.

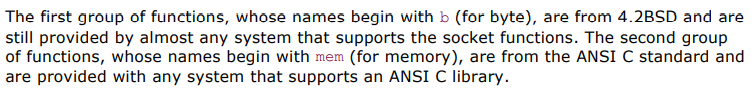


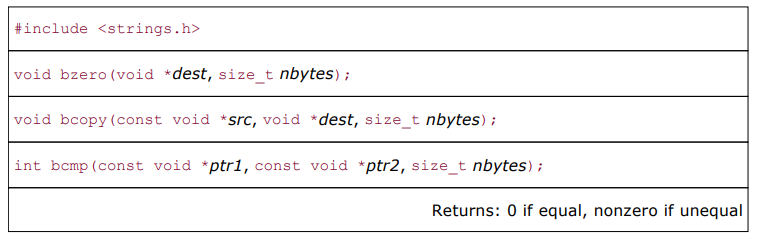


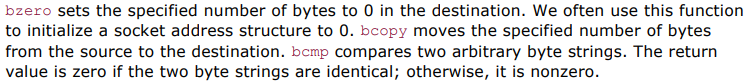


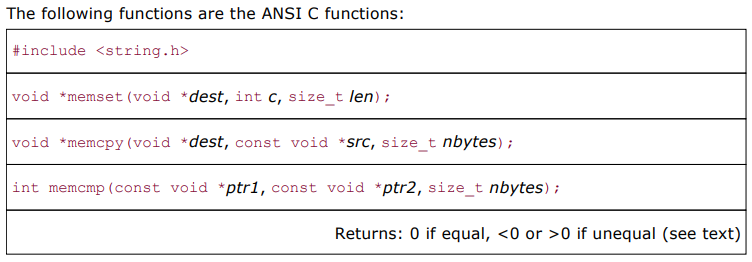
1. Write a note on the Byte Manipulation Functions.



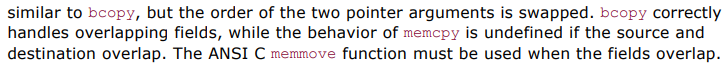


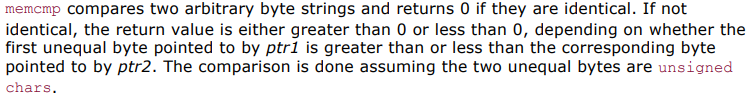




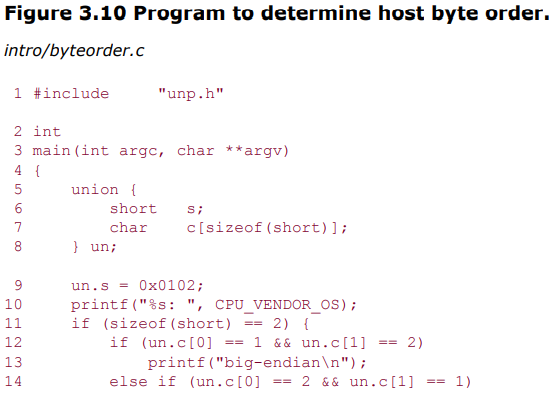


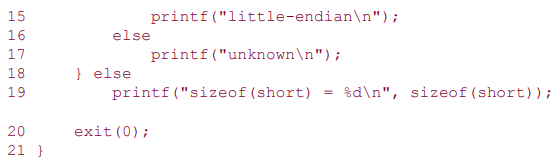




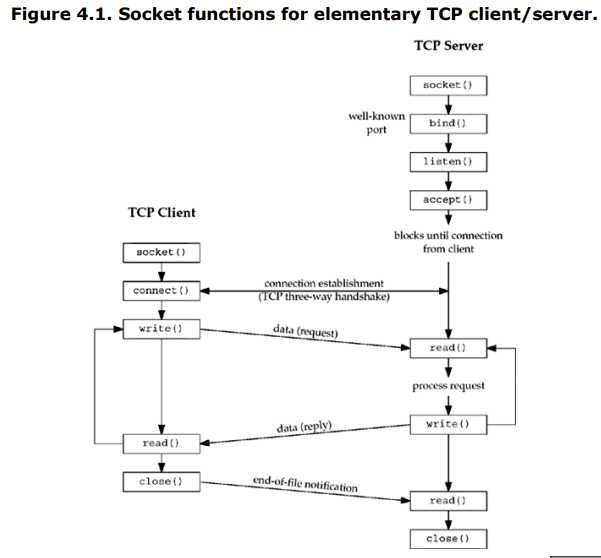


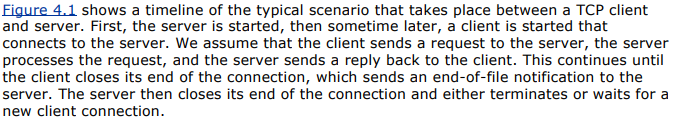
1. Develop a ‘C’ program to determine host byte order.



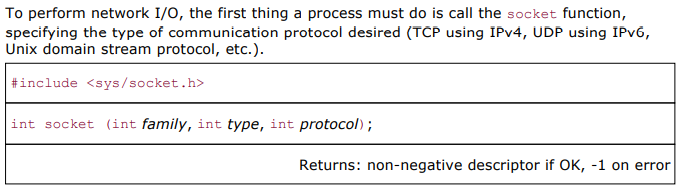


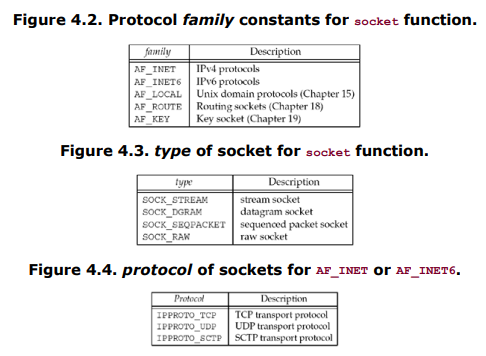
1. Illustrate the significance of socket functions for elementary TCP client/server with a neat block diagram.

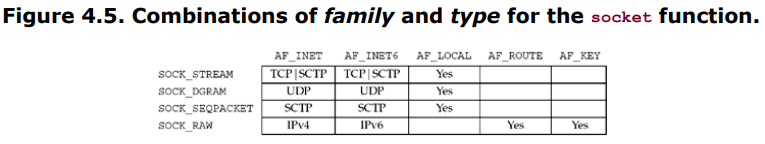




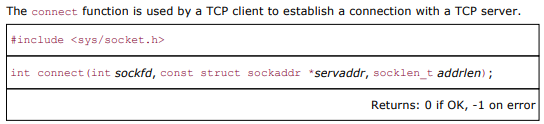
1. Explain the following arguments of the socket function:
   1. Family
   2. Type
   3. Protocol



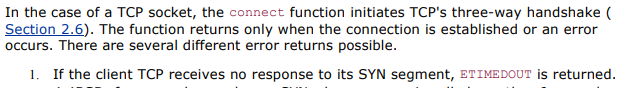


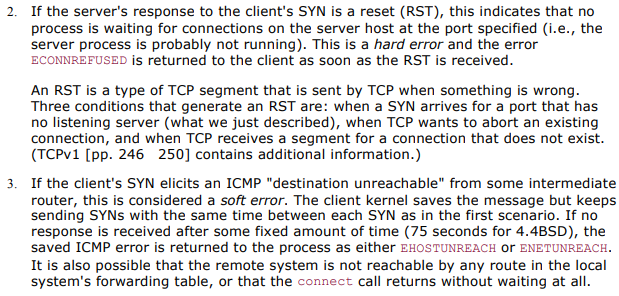


1. Explain the following functions of TCP socket:
   1. Connect

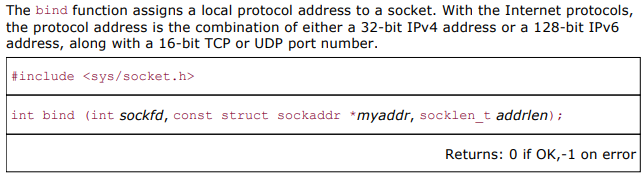


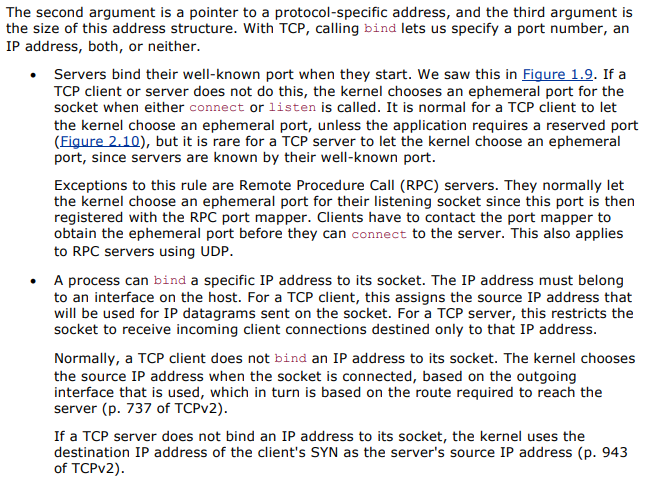






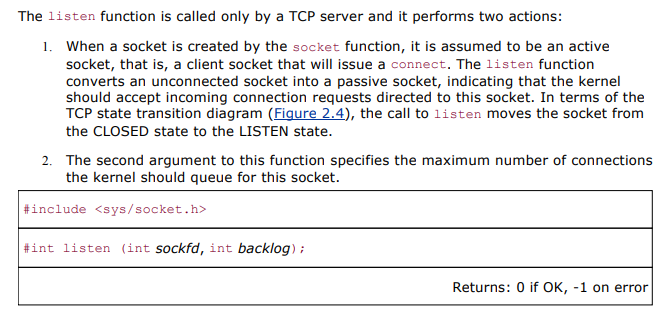
* 1. Bind



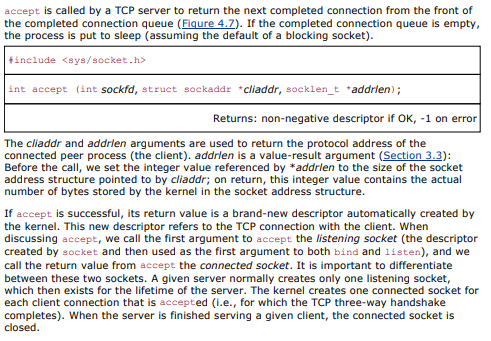




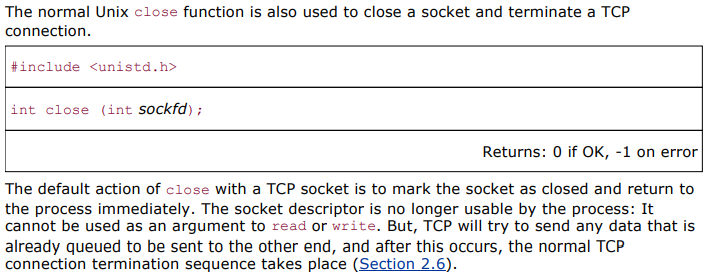
* 1. Listen



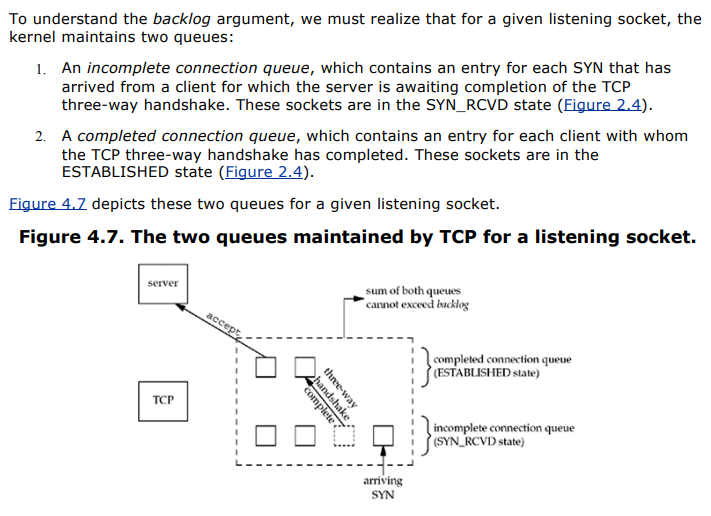
* 1. Accept

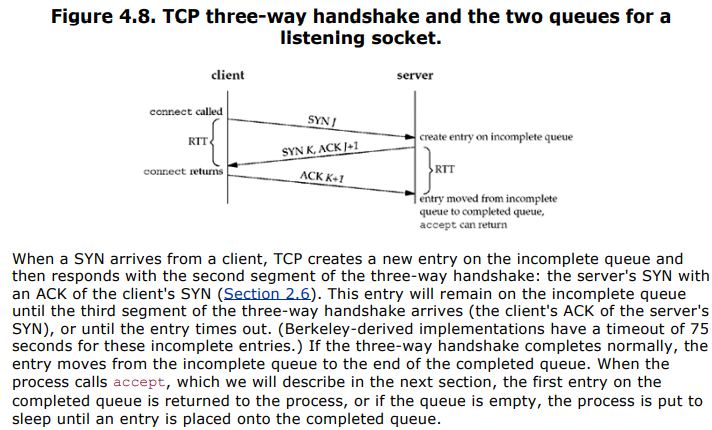


* 1. Close



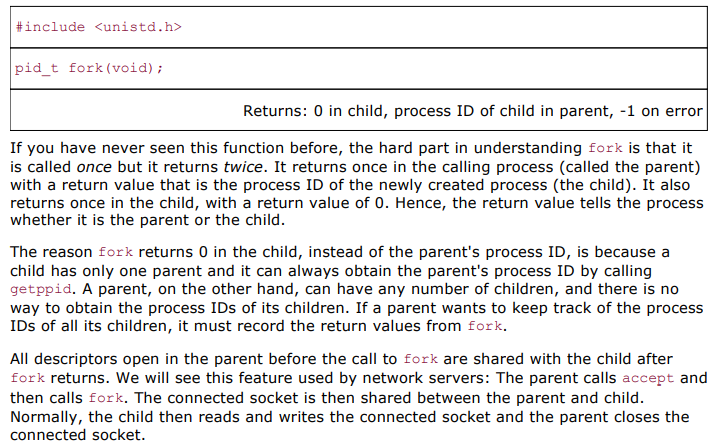
1. With a neat block diagram explain the queues maintained by TCP for a listening socket. Also show the packets exchanged during the connection establishment with these two queues.

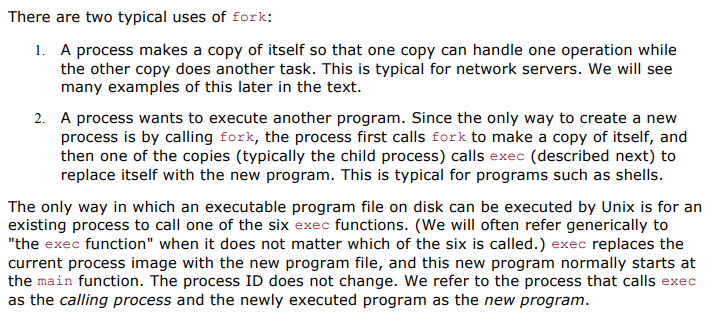


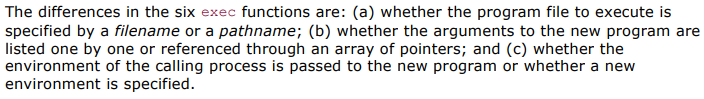


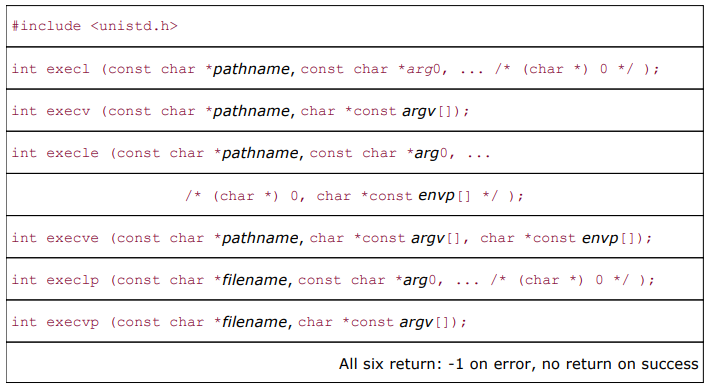
1. Illustrate the significance of fork and exec functions.

This function (including the variants of it provided by some systems) is the only way in Unix to create a new process.

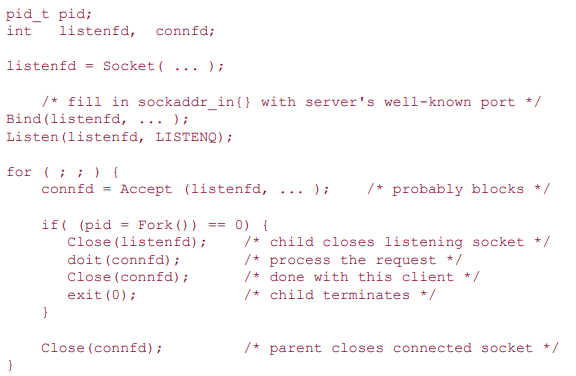




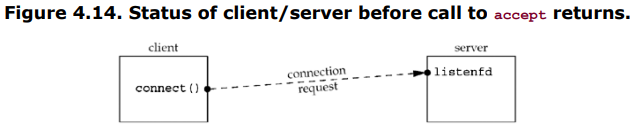


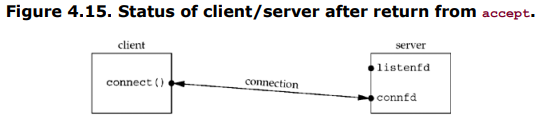


1. Outline the typical concurrent server with the help of pseudocode.

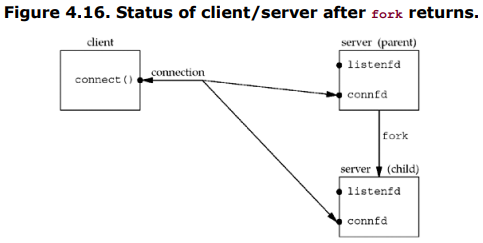


1. Demonstrate the status of client/ server before and after call to *accept* returns with a neat block diagram.

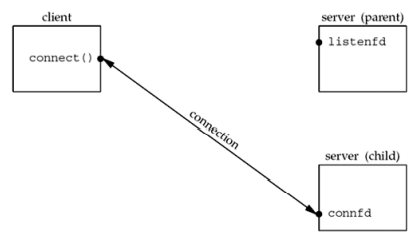




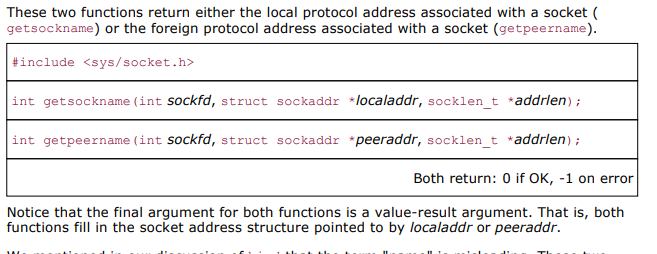
1. Demonstrate the status of client/ server after fork returns with a neat block diagram.

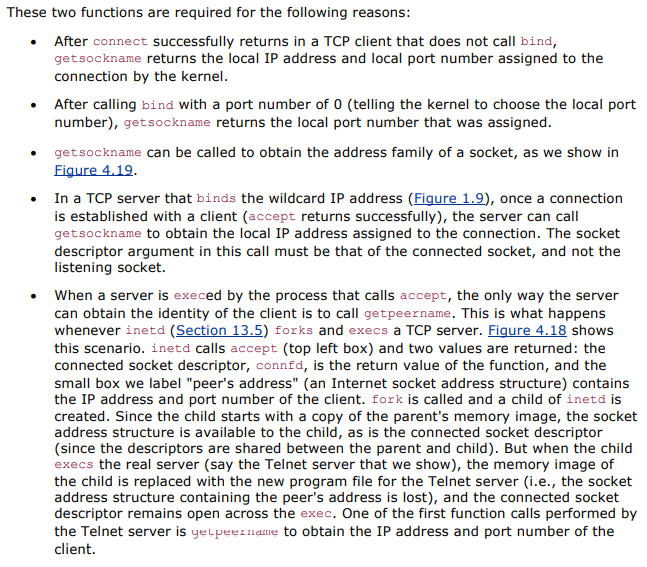


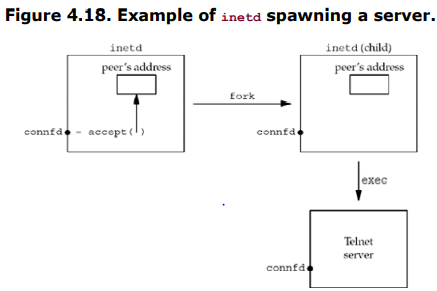
1. Demonstrate the status of client/ server after parent and child close appropriate sockets with a neat block diagram.

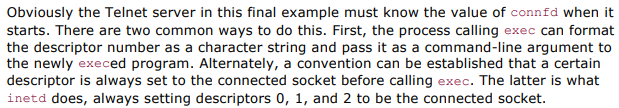


1. Comment on the significance of *getsockname* and *getpeername* functions.

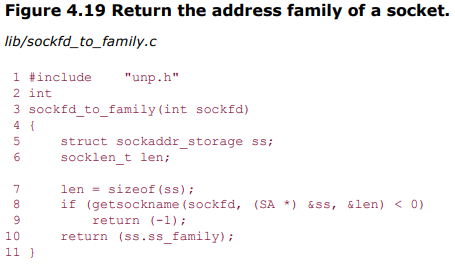






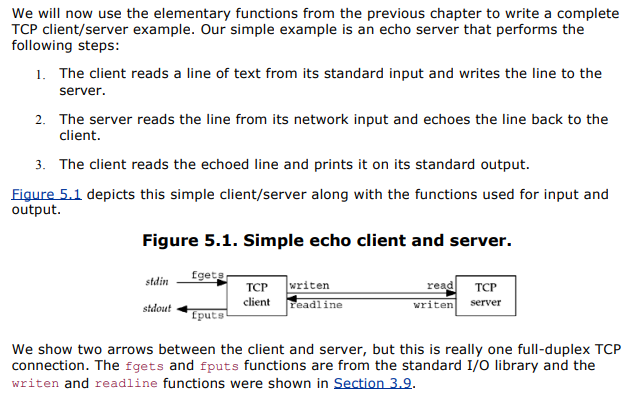


1. Develop the pseudocode that returns the address family of a socket.



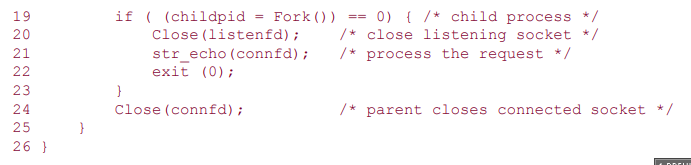
**Self-Learning Topics:**

1. List and explain with a neat block diagram the steps associated with simple TCP echo client and server.

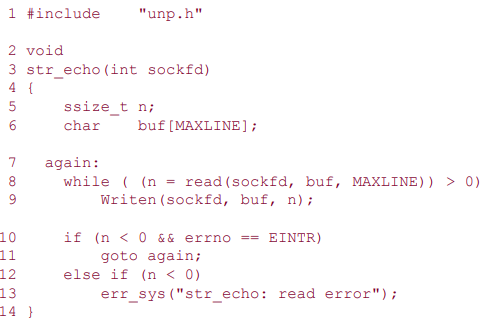


1. Develop the ‘C’ program to demonstrate the TCP echo server: main function





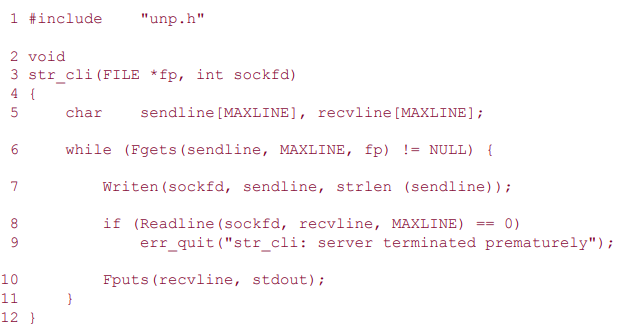
1. Develop the ‘C’ program to demonstrate the TCP echo server: str\_echo function



1. Develop the ‘C’ program to demonstrate the TCP echo client: main function



1. Develop the ‘C’ program to demonstrate the TCP echo client: str\_cli function



**Unit 3: (Elementary UDP Sockets)**

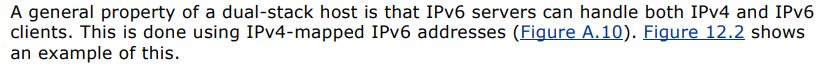
1. Illustrate the significance of socket functions for UDP TCP client/server with a neat block diagram.
2. Explain the following functions of UDP socket:
   1. recvfrom
   2. sendto
3. List and explain with a neat block diagram the steps associated with simple UDP echo client and server.
4. Develop the ‘C’ program to demonstrate the UDP echo server: main function
5. Develop the ‘C’ program to demonstrate the UDP echo server: dg\_echo function
6. Develop the ‘C’ program to demonstrate the UDP echo client: main function
7. Develop the ‘C’ program to demonstrate the UDP echo client: dg\_cli function
8. Outline the summary of TCP client/server with two clients.
9. Outline the summary of UDP client/server with two clients.
10. Develop the ‘C’ program for dg\_cli function that verifies returned socket address.
11. Outline the summary of UDP client/server from client’s perspective with a neat block diagram.
12. Outline the summary of UDP client/server from server’s perspective with a neat block diagram.
13. Develop the ‘C’ program to demonstrate the UDP dg\_cli function that calls connect.
14. Develop the ‘C’ program to demonstrate the UDP dg\_cli function that writes a fixed number of datagrams to the server.
15. Develop the ‘C’ program to demonstrate the UDP dg\_echo function that counts received datagrams.
16. Develop the ‘C’ program to demonstrate the UDP dg\_echo function that increases the size of the socket receive queue.
17. Develop the ‘C’ program for UDP that uses connect to determine outgoing interface.
18. Make use of select function for TCP and UDP Echo server.
19. Illustrate the significance of socket functions for SCTP using one-to-one style with a neat block diagram.
20. Illustrate the significance of socket functions for SCTP using one-to-many style with a neat block diagram.
21. With a neat block diagram explain the shutdown function to close an SCTP association.

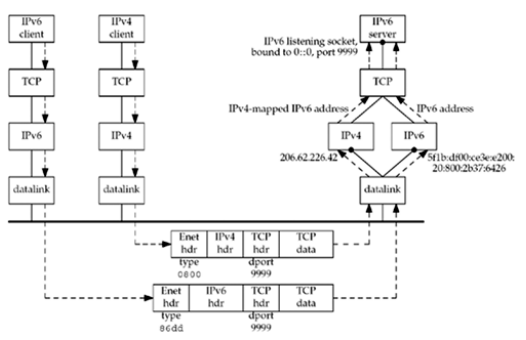
**Self-Learning Topics:**

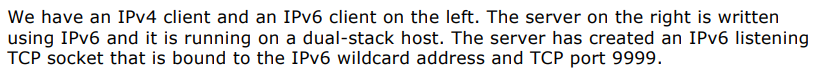
1. Explain the simple SCTP streaming echo client and server with a neat block diagram.

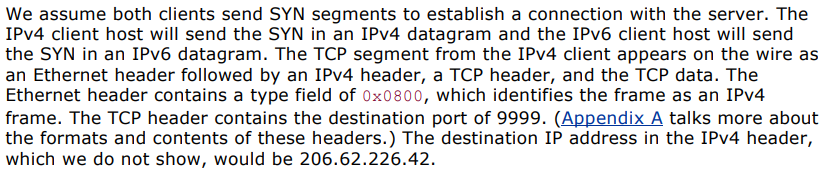
**Unit 4: (Advanced Sockets-I)**

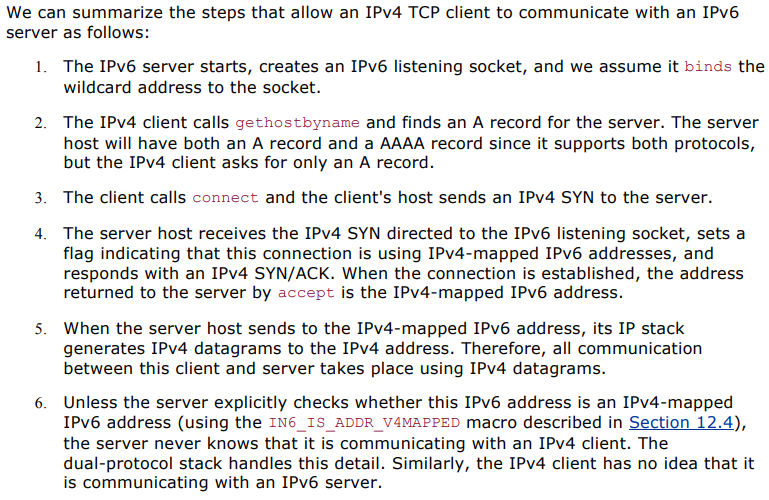
1. With a neat block diagram explain IPv6 server on dual stack host serving IPv4 and IPv6 clients.



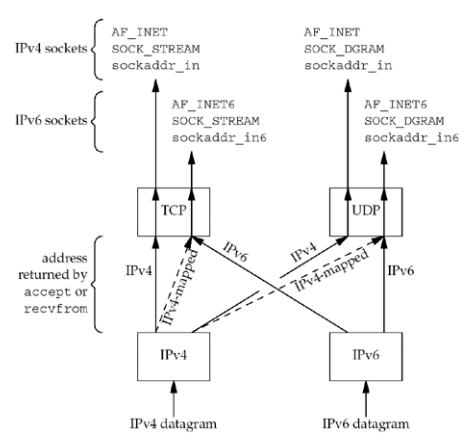


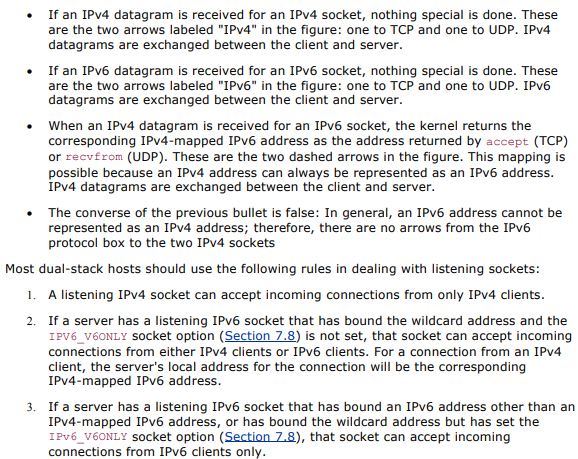




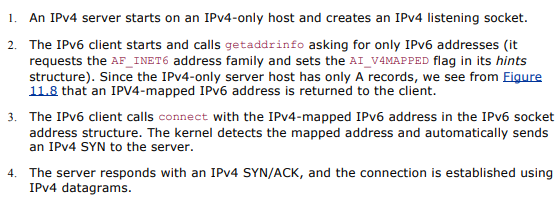


1. Explain with a neat block diagram how the received IPv4 and IPv6 datagrams are processed depending on the type of receiving socket.

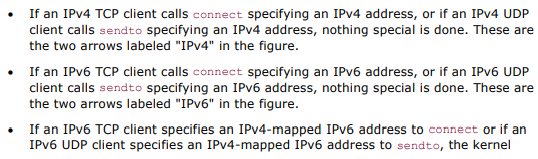


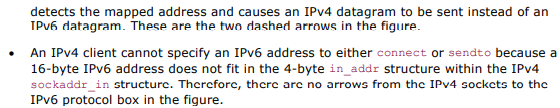


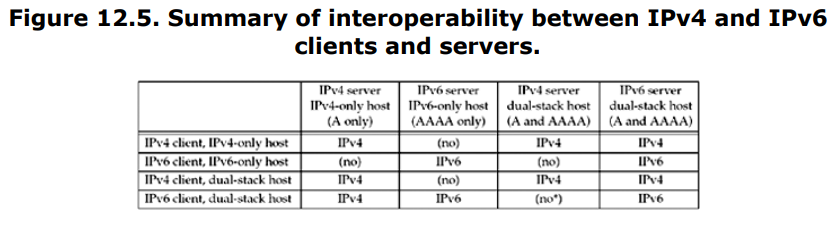
1. Explain with a neat block diagram how the client requests are processed depending on the address type and socket type.





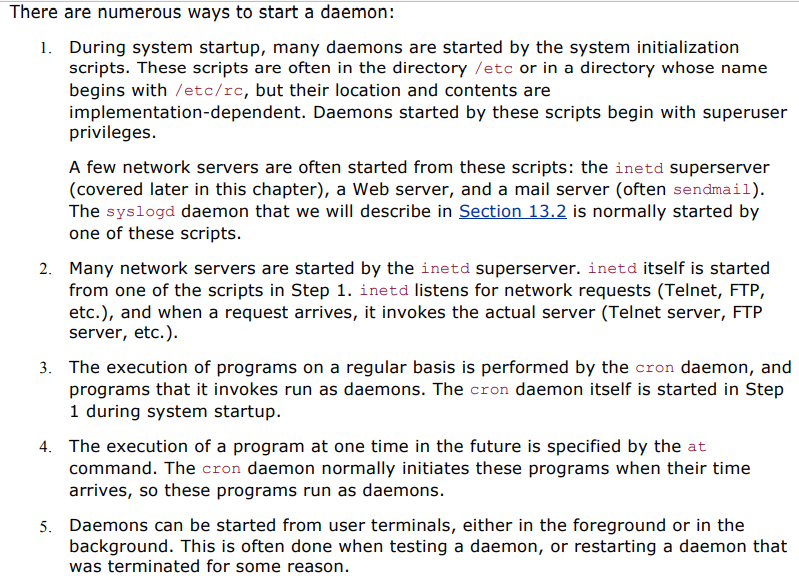




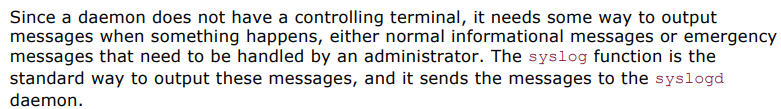


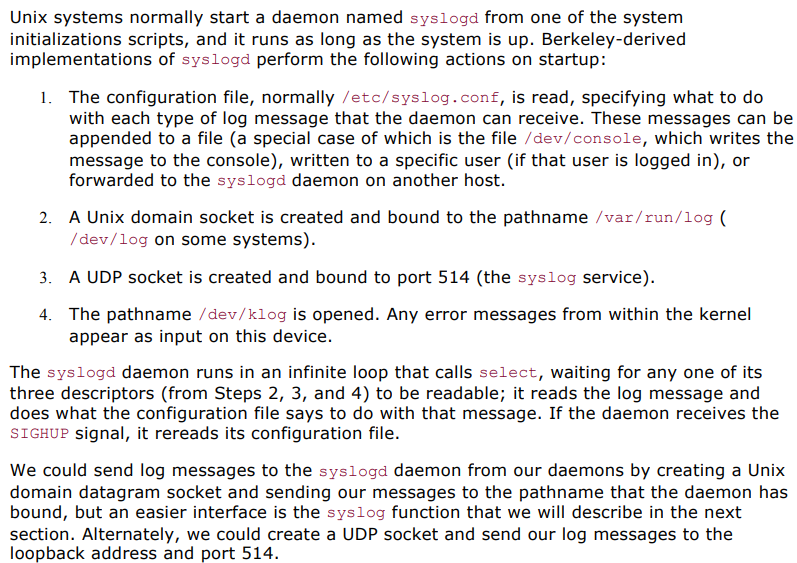
1. List and explain the numerous ways to start a daemon.



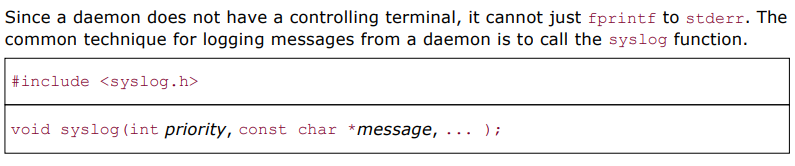


1. List and explain the actions on startup for syslogd Daemon.

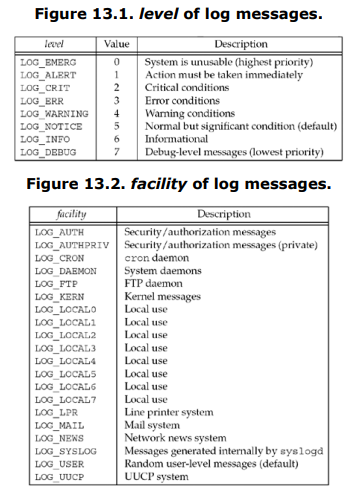


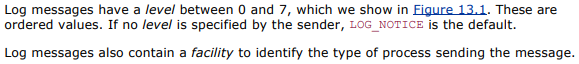


1. With a function prototype explain the syslog Function.







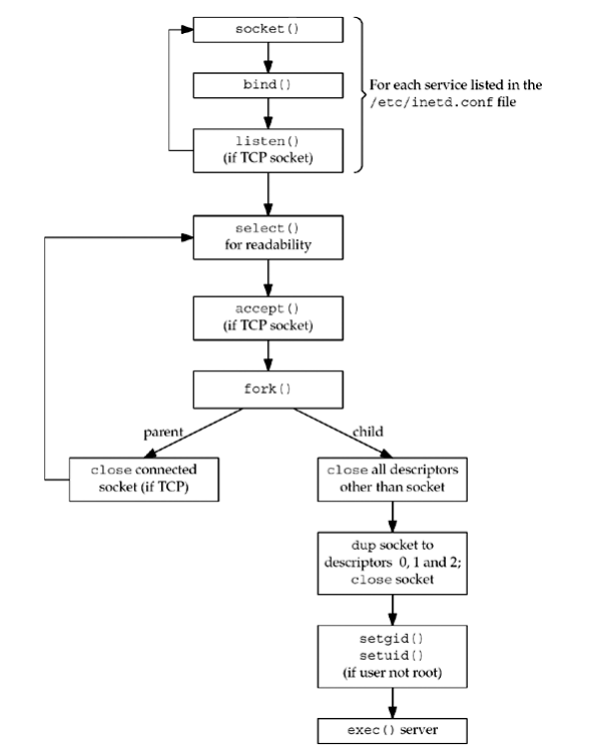


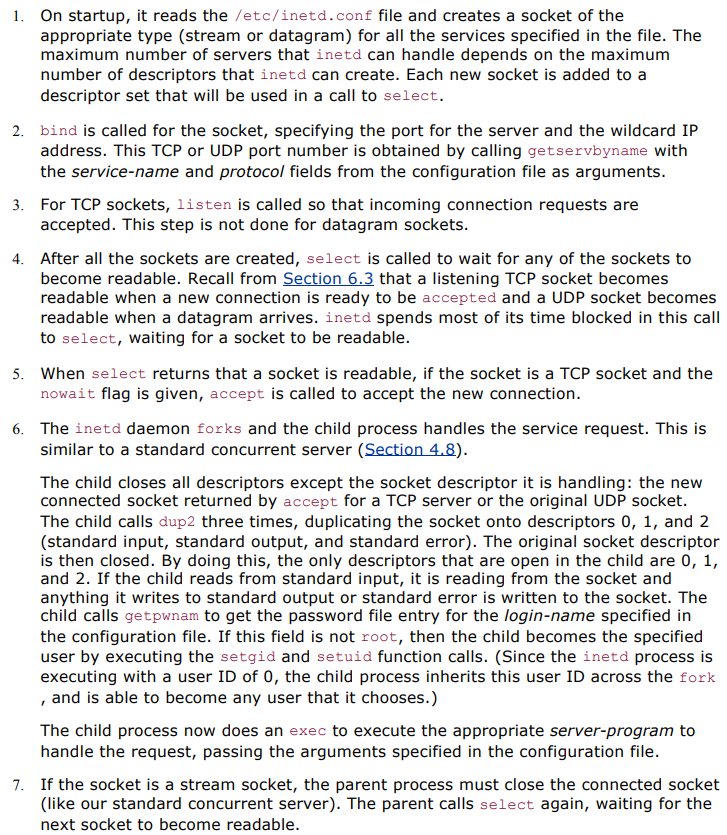




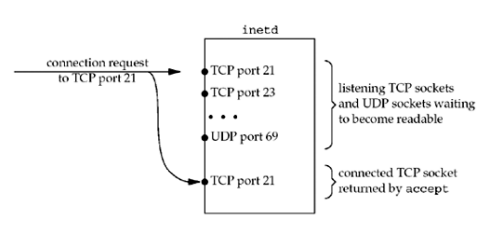
**Self- Learning Topics:**

1. Explain the significance of daemon\_init function.
2. List and explain the steps performed by of inetd Daemon.

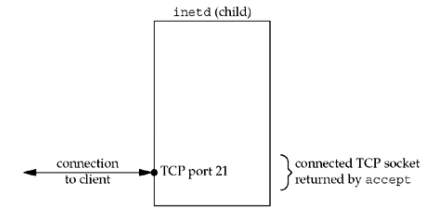




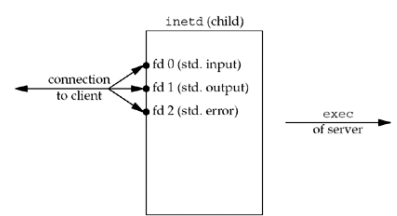
1. With a neat block diagram explain the inetd descriptors when connection request arrives for TCP port.



1. Explain the following with a neat block diagram:
   1. inetd descriptors in child



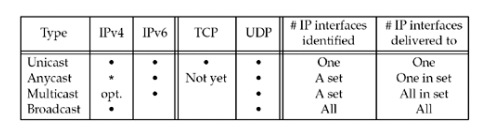
* 1. inetd descriptors after dup2

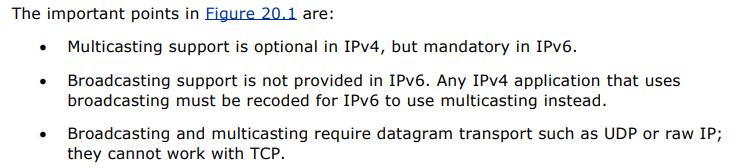


1. Develop the pseudocode for daemon\_inetd function to daemonize process run by inetd.

**Unit 5: (Advanced Sockets-II)**

1. Outline the important points with respect to unicasting, multicasting and broadcasting.

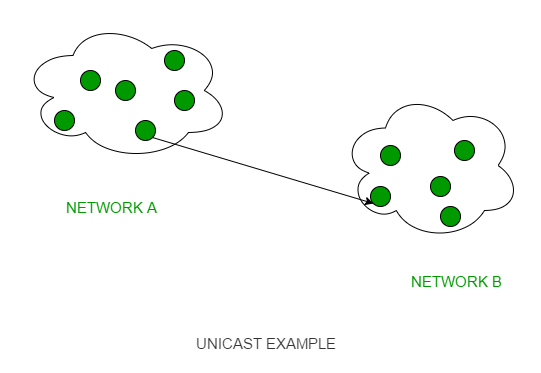




The **cast** term here signifies some data(stream of packets) is being transmitted to the recipient(s) from the client(s) side over the communication channel that helps them to communicate. Let’s see some of the “cast” concepts that are prevailing in the computer networks field.

**1. Unicast –**

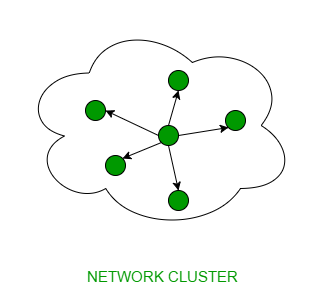
This type of information transfer is useful when there is a participation of a single sender and a single recipient. So, in short, you can term it as a one-to-one transmission. For example, if a device having IP address 10.1.2.0 in a network wants to send the traffic stream(data packets) to the device with IP address 20.12.4.2 in the other network, then unicast comes into the picture. This is the most common form of data transfer over the networks.



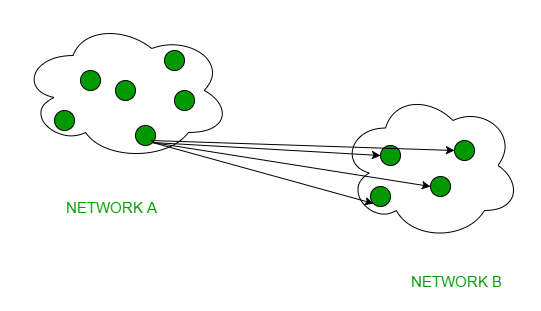
**2. Broadcast –**

Broadcasting transfer (one-to-all) techniques can be classified into two types :

* **Limited Broadcasting –**   
  Suppose you have to send a stream of packets to all the devices over the network that you reside, this broadcasting comes in handy. For this to achieve, it will append 255.255.255.255 (all the 32 bits of IP address set to 1) called as **Limited Broadcast Address** in the destination address of the datagram (packet) header which is reserved for information transfer to all the recipients from a single client (sender) over the network.



* **Direct Broadcasting –**   
  This is useful when a device in one network wants to transfer packet stream to all the devices over the other network. This is achieved by translating all the Host ID part bits of the destination address to 1, referred to as **Direct Broadcast Address** in the datagram header for information transfer.

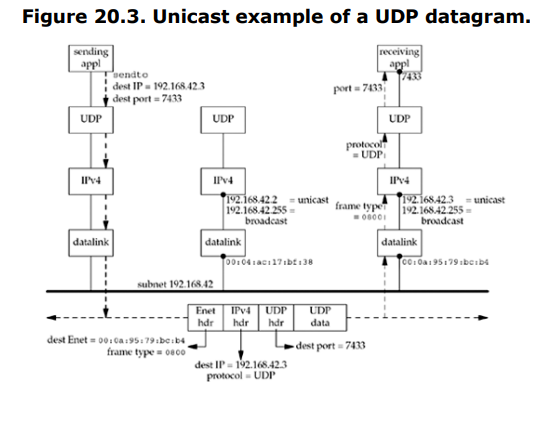


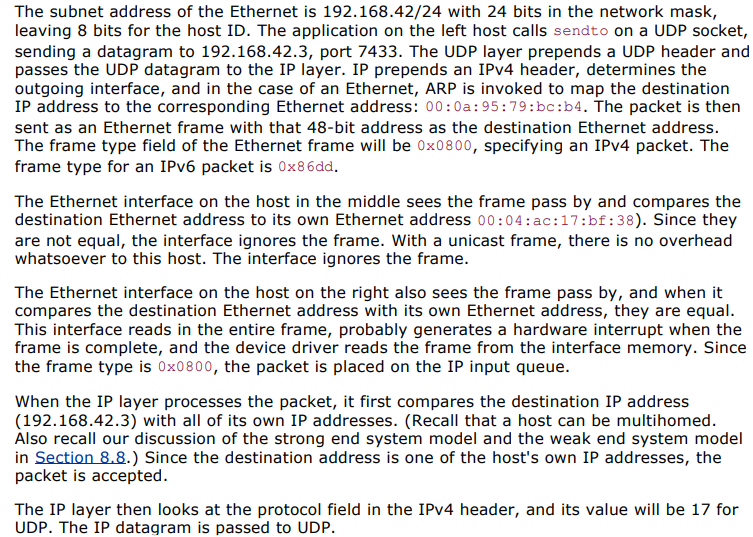
This mode is mainly utilized by television networks for video and audio distribution.   
One important protocol of this class in Computer Networks is [Address Resolution Protocol (ARP)](https://www.geeksforgeeks.org/computer-network-arp-works/) which is used for resolving an IP address into a physical address which is necessary for underlying communication.

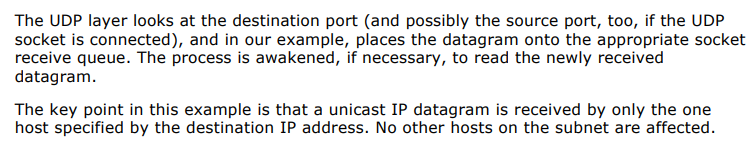
**3. Multicast –**

In multicasting, one/more senders and one/more recipients participate in data transfer traffic. In this method traffic recline between the boundaries of unicast (one-to-one) and broadcast (one-to-all). Multicast lets servers direct single copies of data streams that are then simulated and routed to hosts that request it. IP multicast requires the support of some other protocols like **IGMP (Internet Group Management Protocol), Multicast routing** for its working. Also in Classful IP addressing **Class D** is reserved for multicast groups.

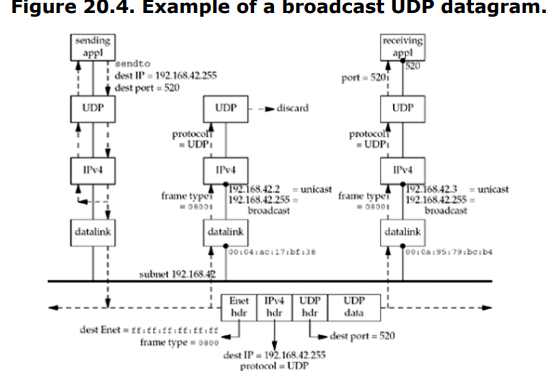
1. List and explain the routing protocols which makes use of multicasting.
2. Differentiate between unicast and broadcast.
3. Make use of UDP datagram to understand unicasting.

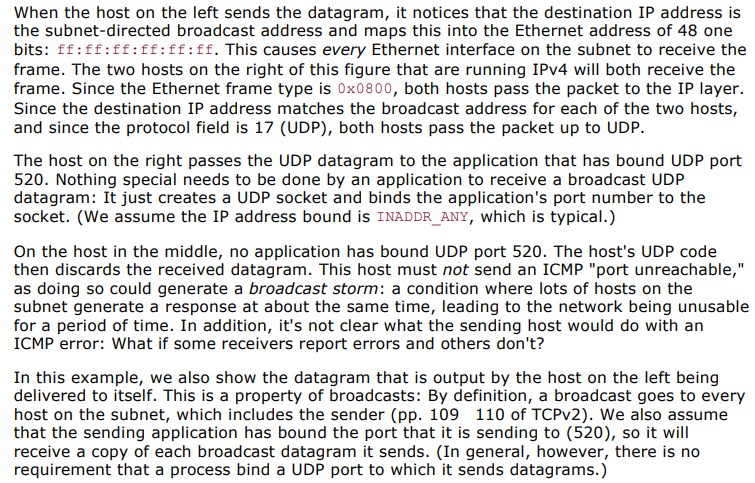


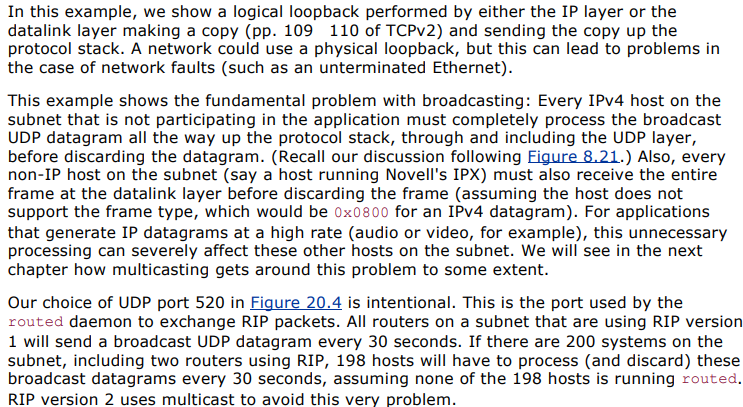




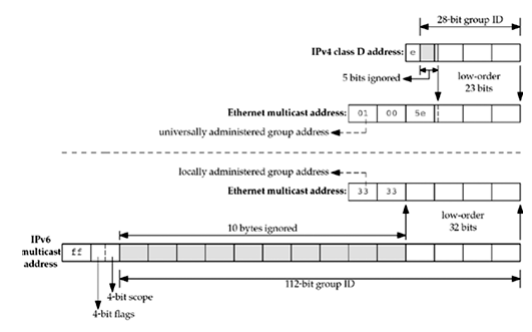
1. Make use of UDP datagram to understand broadcasting.

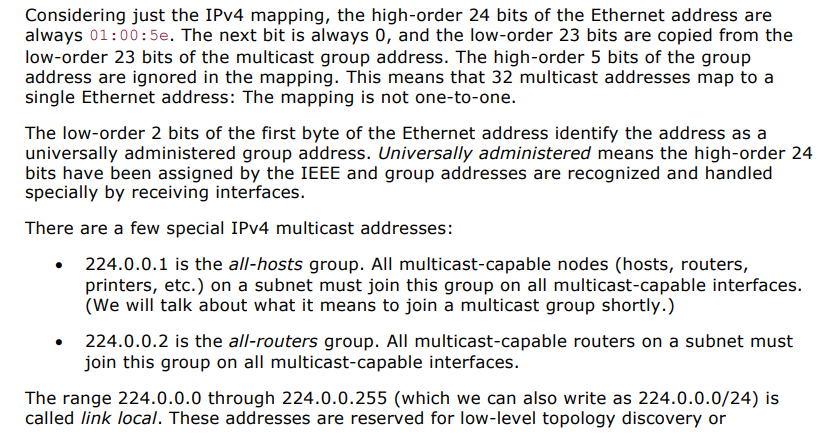


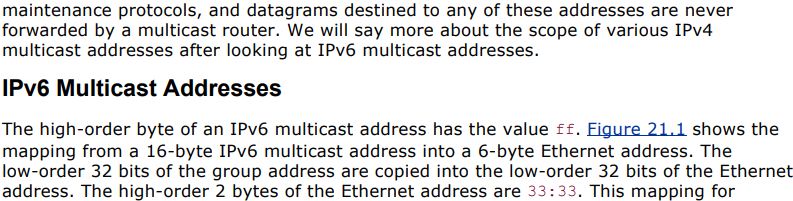


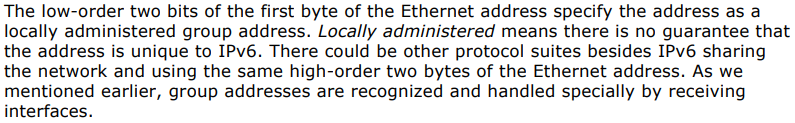


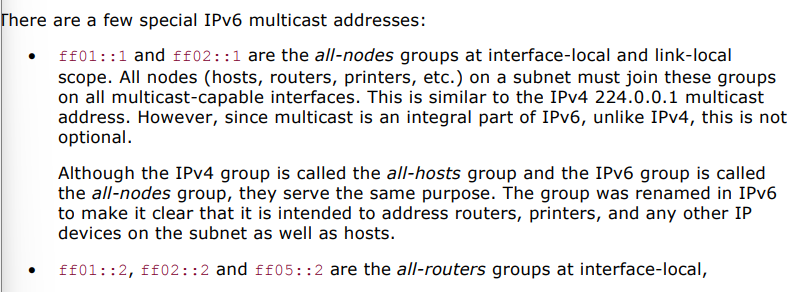
1. Explain with a neat block diagram how IPv4 and IPv6 multicast addresses are mapped to Ethernet addresses.

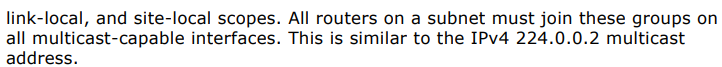




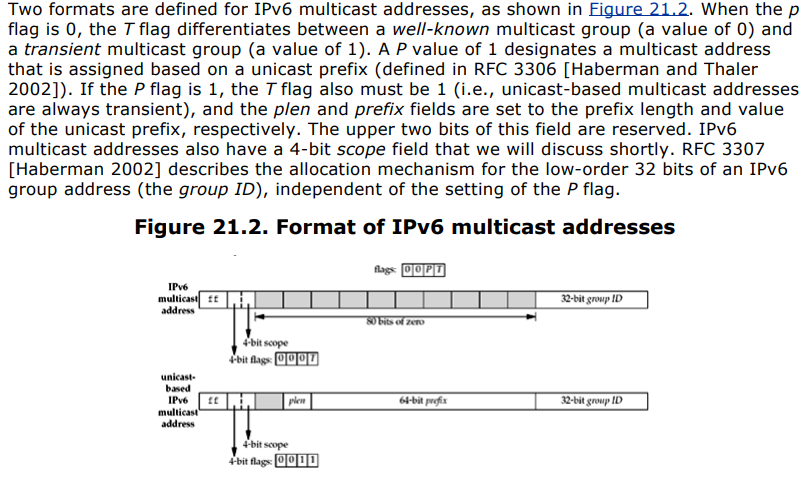




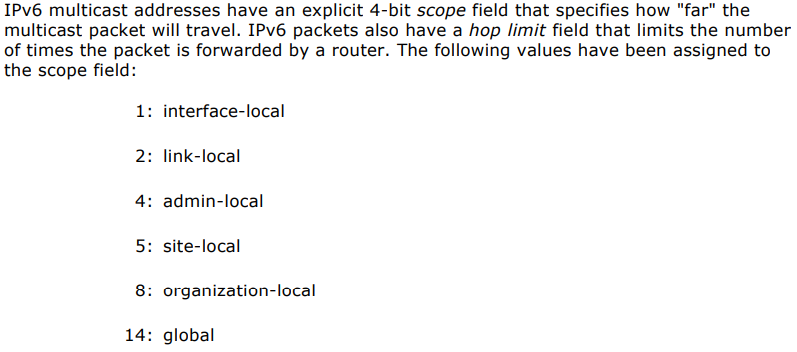


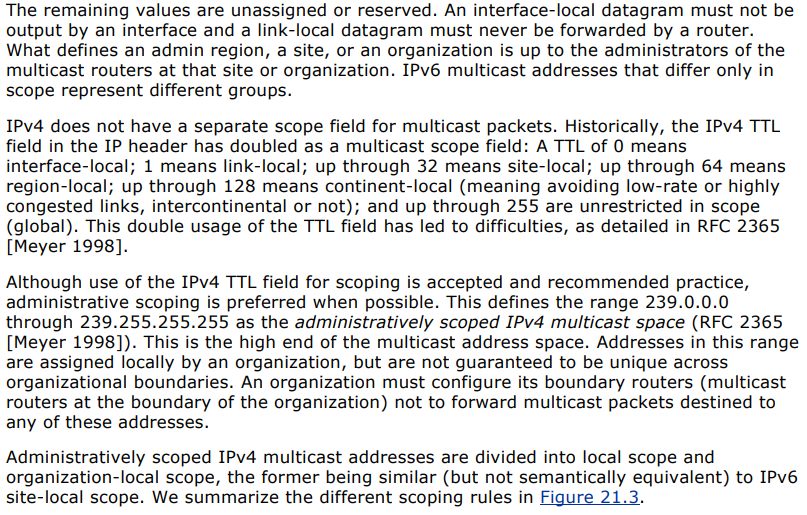


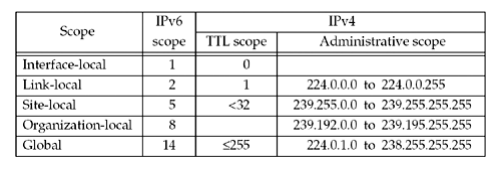
1. What is the format of IPv6 multicast addresses?



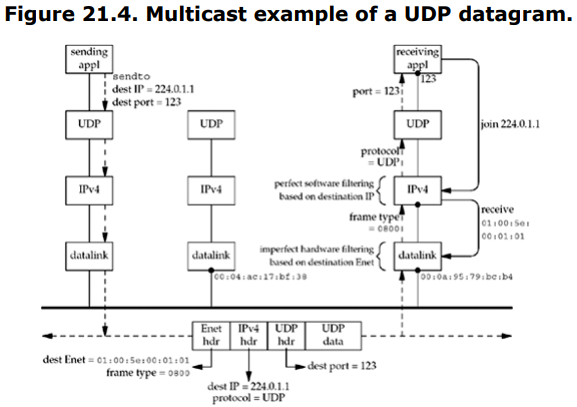
1. Illustrate the scope of multicast addresses.



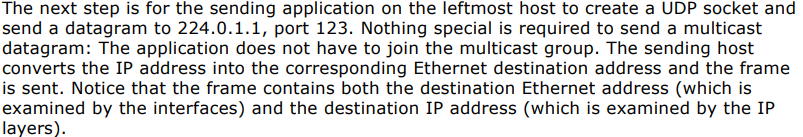


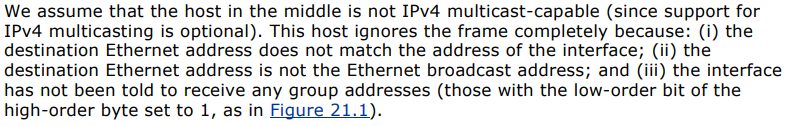


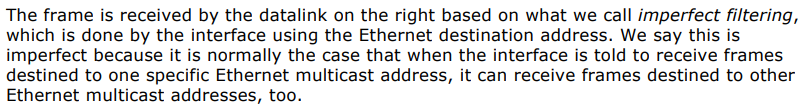
1. Illustrate with a neat block diagram, multicast example of a UDP datagram.

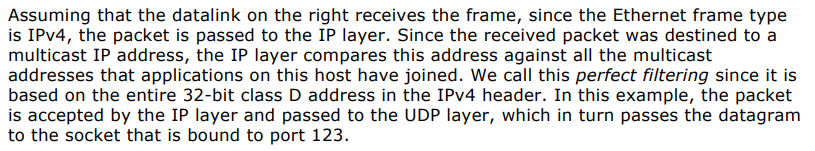


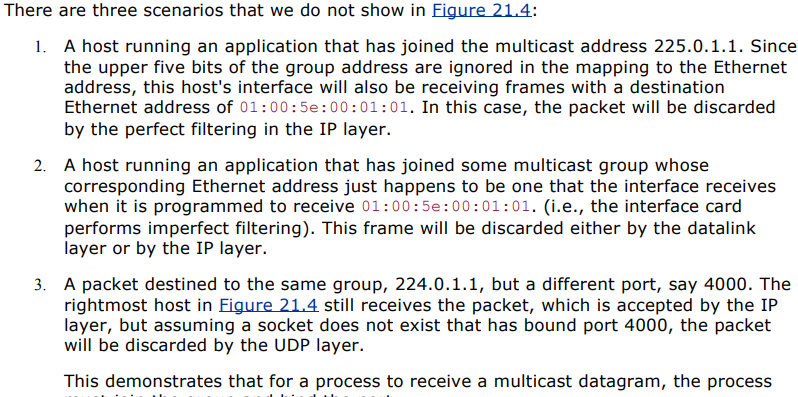






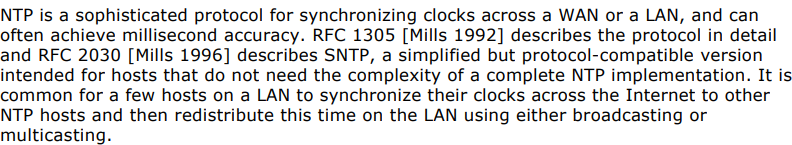




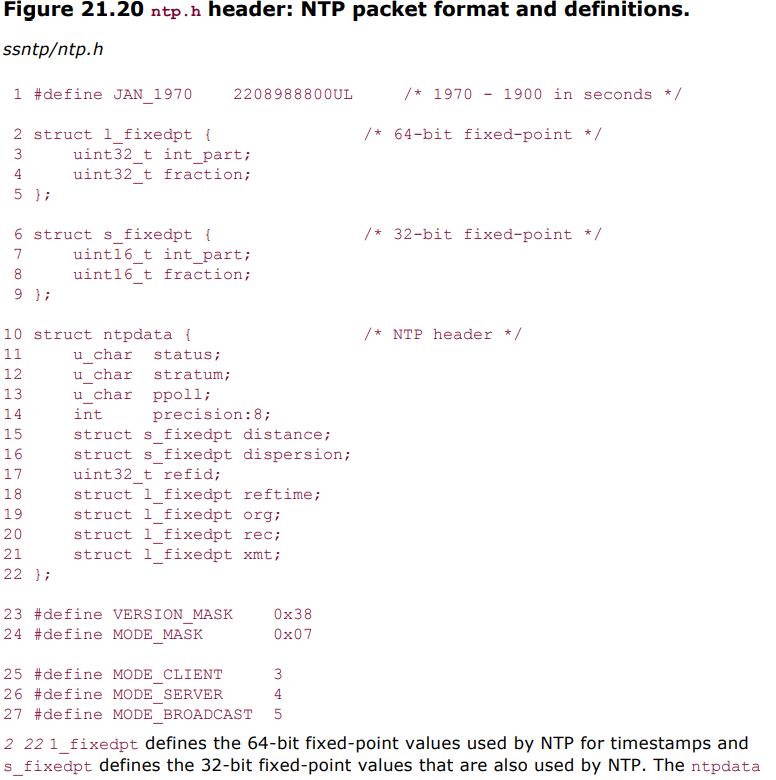




1. Illustrate the NTP packet format and definitions with the help of ntp.h header.









1. Dg\_cli functiom

